Decentralized Eigenvalue Algorithms for Distributed Signal Detection in Cognitive Networks

Federico Penna and Sławomir Stańczak

Abstract

In this paper we derive and analyze two algorithms – referred to as decentralized power method (DPM) and decentralized Lanczos algorithm (DLA) – for distributed computation of one (the largest) or multiple eigenvalues of a sample covariance matrix over a wireless network. The proposed algorithms, based on sequential average consensus steps for computations of matrix-vector products and inner vector products, are first shown to be equivalent to their centralized counterparts in the case of exact distributed consensus. Then, closed-form expressions of the error introduced by non-ideal consensus are derived for both algorithms. The error of the DPM is shown to vanish asymptotically under given conditions on the sequence of consensus errors. Finally, we consider applications to spectrum sensing in cognitive radio networks, and we show that virtually all eigenvalue-based tests proposed in the literature can be implemented in a distributed setting using either the DPM or the DLA. Simulation results are presented that validate the effectiveness of the proposed algorithms in conditions of practical interest (large-scale networks, small number of samples, and limited number of iterations).

Index Terms

Eigenvalue-based signal detection, average consensus, power method, Lanczos algorithm.

The authors are with the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institute, Einsteinufer 37, 10587 Berlin, Germany. Email: {federico.penna, slawomir.stanczak}@hhi.fraunhofer.de. The work was supported by the German Research Foundation (DFG) under grant STA864/3-2 and by the German Federal Ministry of Education and Research under grant 01BU1224. Part of the contents of this paper were presented at the IEEE Globecom 2012 conference [1] and the IEEE Dyspan 2012 conference (poster session) [2].

I. Motivation and Related Work

Computing the eigenvalues of sample covariance matrices is a fundamental problem in signal processing, with applications including multi-sensor spectrum sensing in cognitive radio networks (see Section VI). Given the increasing popularity of dense, large-scale wireless sensor networks, applications of eigenvalue-based inference techniques in distributed settings are of great interest. However, most of the eigenvalue-based techniques proposed in the existing literature assume a centralized architecture, where the samples received by different nodes are forwarded to a fusion center that is in charge of constructing the sample covariance matrix and computing the relevant test statistics. This traditional architecture has several drawbacks: it requires a fusion center with high computational capabilities, therefore it does not support applications in which different nodes may be chosen as fusion centers at different times; relying on one node only, it is vulnerable to hardware failures, Byzantine faults, or attacks from malicious users; it is not efficient in case of multi-hop networks, where some nodes may be many hops away from the fusion center; and it lacks scalability, because a growing number of nodes in the network may result in congestion of the communication channel with the fusion center.

For these reasons, we seek a decentralized method to compute the eigenvalues of sample covariance matrices over a wireless network, such that the computational effort is distributed across multiple nodes and the many-to-one communication protocol is replaced by a more scalable neighbor-to-neighbor protocol. In this paper we propose two solutions based on decentralized implementations of iterative eigenvalue algorithms – the power method (PM, see Section III-B) and the Lanczos algorithm (LA, see Section III-C). Decentralized versions of such algorithms are obtained by applying average consensus (AC, see Section III-A) as a subroutine to perform those computations that involve combining the data of different nodes. Once local estimates of the eigenvalues of interest are computed at every node, statistical tests for signal detection can be performed locally by each node.

The contribution of this paper is related to that of [3], where a decentralized algorithm based on the Oja-Karhunen recursion is proposed to track the eigenvectors of a covariance matrix. Our work adopts the same idea of computing inner vector products through AC (and further extends this approach to matrix-vector products), but differs from [3] in the following sense: *(i)* we compute eigenvalues instead of (possibly, in addition to) eigenvectors, thus adapting the

well-established theory of eigenvalue-based detection to decentralized network settings; *(ii)* we focus on detection (decision based on N received samples per sensor) instead of sequential tracking (update of eigenvector estimates at every new sample). This makes our approach suitable for spectrum sensing and other signal detection applications. A similar methodology for distributed matrix multiplication via AC has been recently used also in [4], where a decentralized expectation-maximization algorithm is derived for static linear Gaussian models.

Other recent related works are [5] and [6], both proposing decentralized implementations of principal component analysis (PCA) over wireless networks. The first approach [5] relies on the assumption of decomposable Gaussian graphical models, i.e., it requires data sample to be *(i)* multivariate Gaussian distributed, and *(ii)* decomposable into two or more conditionally independent "cliques". The global eigenvalue decomposition (EVD) problem is thus broken down into a sequence of local (clique-wise) EVD subproblems. The second approach [6] combines the power method with the concept of sparsification to achieve an efficient distributed computation of the eigenvectors and eigenvalues of a symmetric matrix. However, this approach is based on the assumptions that *(i)* each node has access to a full row of the matrix (which is not the case with the sample covariance matrix considered in our model, see Section II), and *(ii)* the network graph is completely connected (i.e., a direct link exists between any pair of nodes). Our approach, on the contrary, does not require any of the aforementioned assumptions.

The paper is organized as follows. A formal statement of the problem is provided in Section II. Section III contains mathematical preliminaries about the algorithms used in this work (AC, PM, and LA). We then introduce the proposed algorithms in Section IV and analyze their performance and complexity in Section V. We finally discuss two practical applications of the proposed algorithms in Section VI and present numerical results in Section VII. Concluding remarks are provided in Section VIII.

II. Problem Statement

Consider a wireless network consisting of K sensor nodes. During a given time interval (sensing period), each node collects N complex signal samples. The global received sample

matrix is denoted by

$$\mathbf{Y} = [\mathbf{y}(1), \ldots, \mathbf{y}(N)] = \begin{bmatrix} \mathbf{y}[1]^T \\ \vdots \\ \mathbf{y}[K]^T \end{bmatrix} \in \mathbb{C}^{K \times N}, \tag{1}$$

where symbols $\mathbf{y}(\cdot) \in \mathbb{C}^K$ and $\mathbf{y}[\cdot] \in \mathbb{C}^N$ are used to denote, respectively, the columns and (transpose) rows of $\mathbf{Y}$. Physically, column $\mathbf{y}(n)$ contains the samples received by all nodes at time n, while row $\mathbf{y}[k]^T$ contains all samples available at node k at the end of the sensing period. We then define the sample covariance matrix as

$$\mathbf{R} \triangleq \frac{1}{N}\mathbf{Y}\mathbf{Y}^H. \tag{2}$$

Let $\lambda_1 \geq \ldots \geq \lambda_K \geq 0$ be the eigenvalues of $\mathbf{R}$, without loss of generality sorted in decreasing order, and $\mathbf{u}_1, \ldots, \mathbf{u}_K$ the corresponding eigenvectors. The problem addressed in this work can be stated as follows: how can a network compute (or estimate) one or more of the above eigenvalues without a fusion center that collects all samples $\mathbf{Y}$, and without explicitly constructing the sample covariance matrix $\mathbf{R}$? Before presenting the proposed solution, we introduce some preliminary definitions and basic concepts about distributed consensus, power method, and Lanczos algorithm. The notation used throughout the paper is summarized in Table I.

III. Preliminaries

A. Average Consensus

Assume that the network nodes and their links form a connected undirected graph. Under such an assumption, it is possible to define a distributed AC algorithm over the network. By distributed AC we mean any algorithm whose output for all nodes converges to the average of the initial values of the individual nodes. A large variety of AC algorithms have been proposed in the literature (see [7] for a survey), both with synchronous [8], [9] and asynchronous protocols [10]–[12]. Extensions to noisy message exchange and link failures include [13]–[17], and methods for consensus acceleration have been proposed in [18]–[20]. It is worth noting that, under the assumption of fixed network topology and noiseless communication, exact consensus can be achieved in a finite number of steps: see for example [21].

Symbol	Definition
$\|\mathbf{v}\|$	Euclidean (ℓ^2) norm of vector $\mathbf{v}$
^{T}, *, ^{H}	Transpose, complex conjugate, and Hermitian operators
$\mathbf{a}(n)$, $\mathbf{a}[k]^T$	Respectively, column n and row k of matrix $\mathbf{A}$
$\odot$, $\oslash$	Element-wise vector multiplication and division
$\mathbf{I}_m$, $\mathbf{1}_m$	Identity matrix and column vector of ones of size m (subscript is omitted when not ambiguous)
$\mathrm{diag}(\mathbf{A})$	Column vector with diagonal elements of a square matrix $\mathbf{A}$: i.e., $\mathrm{diag}(\mathbf{A}) = (\mathbf{A} \odot \mathbf{I}) \cdot \mathbf{1}$
$\mathrm{Diag}(\mathbf{v})$	Square matrix with vector $\mathbf{v}$ as main diagonal: i.e., $\mathrm{Diag}(\mathbf{v}) = (\mathbf{v} \cdot \mathbf{1}^T) \odot \mathbf{I}$
$\mathtt{RN}(\mathbf{A})$	Vector of ℓ^2 square norms of the rows of $\mathbf{A}$: i.e., $\mathtt{RN}(\mathbf{A}) \triangleq \left[\|\mathbf{a}[1]\|^2, \ldots, \|\mathbf{a}[K]\|^2\right]^T = (\mathbf{A}^* \odot \mathbf{A}) \cdot \mathbf{1}$
$\mathtt{AC}_m^t(\cdot)$	AC function with global input/output for all nodes (Eq. 4) m = input vector size, t = AC iterations or running time (no superscript = ideal AC)
$\mathtt{AC}_m^t[k](\cdot)$	AC function with local input/output for node k (Eq. 6)

TABLE I
NOTATION.

In this paper we do not adopt a specific AC algorithm, but rather take a general approach. We model the result of a generic AC routine by a function $\mathtt{AC}_m^t : \mathbb{C}^{K\times m} \to \mathbb{C}^{K\times m}$, where m is the size of the input vectors at each node[1] and t is the number of iterations or the averaging time.[2] Denote the global input (or initial value) matrix by

$$\mathbf{Z}_0 = [\mathbf{z}_0(1), \ldots, \mathbf{z}_0(m)] = \begin{bmatrix} \mathbf{z}_0[1]^T \\ \vdots \\ \mathbf{z}_0[K]^T \end{bmatrix} \in \mathbb{C}^{K\times m}, \tag{3}$$

defined in analogy with (1), such that the k-th row represents the samples available at node k. Then, the output of the AC function at time t is defined by

$$\mathbf{Z}_t = \mathtt{AC}_m^t(\mathbf{Z}_0) \triangleq \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathbf{Z}_0 + \mathbf{E}_t, \tag{4}$$

[1]An AC algorithm with vector inputs involves exchanging m scalar numbers at every iteration, and returns the average of the input vectors over all network nodes. As such, it involves the same number of messages as scalar AC, but with a larger payload.

[2]For sake of generality, we do not specify whether the adopted AC routine should be synchronous or asynchronous. In the first case t is a discrete number of iterations; in the second case it is the running time of the algorithm, that is related probabilistically to the number to clock ticks (see for example [11]).

where $\mathbf{1}$ is a column vector of ones of size K, and

$$\mathbf{E}_t = [\mathbf{e}_t(1), \ldots, \mathbf{e}_t(m)] \in \mathbb{C}^{K \times m} \tag{5}$$

is an error term depending on the AC time t and on the specific AC method. In general, $\mathbf{E}_t$ can be assumed to be either bounded (at least statistically, see [9]–[12], [14]–[16], [19]) or equal to zero (in the case of finite-time AC [21]).

In the following, it is sometimes convenient to express the input and the output of AC for a single node k. For this purpose, we define a function $\mathtt{AC}_m^t[k] : \mathbb{C}^{1\times m} \to \mathbb{C}^{1\times m}$ having as input/output the k-th rows of the input/output matrices of the global function $\mathtt{AC}_m^t(\cdot)$ defined in (4). Thus, we can write

$$\mathbf{z}_t[k]^T = \mathtt{AC}_m^t[k](\mathbf{z}_0[k]^T). \tag{6}$$

When AC is applied to scalar arguments ($m = 1$), the global input-output relation is written as $\mathbf{z}_t = \mathtt{AC}_1^t(\mathbf{z}_0)$, with $\mathbf{z}_0, \mathbf{z}_t \in \mathbb{C}^K$. Finally, if $\mathbf{E}_t = \mathbf{0}$ in (4), we call the AC routine "ideal" and we use the notation $\mathtt{AC}_m(\cdot)$ (without superscript).

B. Power Method

The PM is a well-known iterative algorithm that, given a square matrix, converges to the eigenvector associated with the largest eigenvalue of the matrix [22]. The iteration, applied to the sample covariance matrix $\mathbf{R}$, can be written as

$$\mathbf{v}_{(j+1)} = \mathbf{R}\mathbf{v}_{(j)}, \tag{7}$$

where $\mathbf{v}_{(0)} \in \mathbb{C}^K$ is an arbitrary starting vector. By (7), the j-th iteration can be written as $\mathbf{v}_{(j)} = \mathbf{R}^j\mathbf{v}_{(0)}$ and, for $j \to \infty$, the vector $\mathbf{v}_{(j)}$ converges to a multiple of the eigenvector $\mathbf{u}_1$. The convergence rate of the algorithm is $O((\lambda_2/\lambda_1)^M)$ [22]. After M iterations, the largest eigenvalue of $\mathbf{R}$ can be approximated by

$$\hat{\lambda}_1 = \frac{\mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)}}{\mathbf{v}_{(M)}^H \mathbf{v}_{(M)}}. \tag{8}$$

C. Lanczos Algorithm

A more sophisticated eigenvalue algorithm is the LA, originally proposed by Lanczos in [23]. The LA is applicable only to symmetric or Hermitian matrices (which is the case of the sample covariance matrix $\mathbf{R}$ considered here), but provides estimates of multiple eigenvalues (the number

of estimated eigenvalues depends on the iterations of the algorithm) and has faster convergence than the PM [24, Chapter 6]. The advantage of the LA over the PM lies in the fact that the LA takes into account, at every iteration j, the complete Krylov subspace

$$\mathcal{K}_j(\mathbf{R}, \mathbf{v}_{(0)}) = \mathrm{span}\{\mathbf{v}_{(0)}, \mathbf{R}\mathbf{v}_{(0)}, \ldots, \mathbf{R}^j\mathbf{v}_{(0)}\} \tag{9}$$

whereas the PM only considers the last term $\mathbf{R}^j\mathbf{v}_{(0)}$. The LA has been thoroughly studied by Paige [25]–[27]. In particular, several computational variants are compared in [25], showing that some are numerically more stable than others. Among the "stabler" variants, we adopt the one named "A(1,7)" in [25] or equivalently "A2" in [26]. The same version of the algorithm is presented in [28, p. 651]. This variant is convenient in view of the decentralized implementation that will be developed in Section IV-B.

The derivation of the algorithm can be briefly outlined as follows. Due to the Hermitian structure of $\mathbf{R}$, for a given $M \leq K$, we can write

$$\mathbf{R}\mathbf{V} = \mathbf{V}\mathbf{T} \tag{10}$$

where the columns of $\mathbf{V} \in \mathbb{C}^{K\times M}$ are mutually orthogonal unit-norm vectors and $\mathbf{T} \in \mathbb{C}^{M\times M}$ is a tridiagonal matrix. If $M = K$, matrices $\mathbf{T}$ and $\mathbf{R}$ are similar, so their eigenvalues are the same. However, as Lanczos first noted, the eigenvalues of $\mathbf{T}$ (sometimes referred to as "Ritz values") turn out to be excellent approximations of the eigenvalues of $\mathbf{R}$ even when $M < K$. The LA is thus defined by iteratively equating the columns of $\mathbf{RV}$ to those of $\mathbf{VT}$. If we let

$$\mathbf{T} = \begin{bmatrix} \alpha_{(1)} & \beta_{(2)} & & & \\ \beta_{(2)} & \alpha_{(2)} & \beta_{(3)} & & \\ & \cdots & \cdots & \cdots & \\ & & \beta_{(M-1)} & \alpha_{(M-1)} & \beta_{(M)} \\ & & & \beta_{(M)} & \alpha_{(M)} \end{bmatrix}, \tag{11}$$

the j-th iteration of the LA can be written as

$$\alpha_{(j)} = \mathbf{v}_{(j)}^H \mathbf{R}\mathbf{v}_{(j)} \tag{12}$$

$$\mathbf{w}_{(j)} = \mathbf{R}\mathbf{v}_{(j)} - \alpha_{(j)}\mathbf{v}_{(j)} - \beta_{(j)}\mathbf{v}_{(j-1)} \tag{13}$$

$$\beta_{(j+1)} = \|\mathbf{w}_{(j)}\| \tag{14}$$

$$\mathbf{v}_{(j+1)} = \mathbf{w}_{(j)}/\beta_{(j+1)}, \tag{15}$$

with an arbitrary starting vector $\mathbf{v}_{(0)}$ of unit norm, and $\beta_{(1)} = 0$. The above iteration is repeated for $j = 1$ to M, thus obtaining the coefficients $\alpha_{(j)}$ and $\beta_{(j)}$ which are necessary to construct $\mathbf{T}$. The desired estimates $\hat{\lambda}_1, \ldots, \hat{\lambda}_M$ of the eigenvalues of $\mathbf{R}$ are then set to be the eigenvalues of $\mathbf{T}$. Note that the eigenvalues of a tridiagonal matrix of size $M \times M$ can be efficiently computed with complexity $O(M^2)$ by using the spectral bisection method [29].

IV. Proposed Algorithms

We now investigate how the aforementioned PM and LA can be implemented in a distributed fashion over a wireless network. That is, the goal is for each node $k \in \{1, 2, \ldots, K\}$ to compute local estimates $\{\hat{\lambda}_i[k]\}$ of the eigenvalue(s) of interest, namely, $i = 1$ for the PM and $1 \leq i \leq M$ for the LA. Decentralized eigenvalue estimates should be as close as possible to their centralized counterparts: ideally, for every eigenvalue λ_i of interest, we would like to have

$$\hat{\lambda}_i[k] = \hat{\lambda}_i \quad \forall k. \tag{16}$$

In the following sections we show that efficient DPM and DLA schemes can be developed by distributing the PM and LA vector iterations – respectively, (7) and (12)-(15) – in such a way that the k-th element of vector $\mathbf{v}_{(j)}$, indicated by $v_{(j)}[k]$, is computed by node k. This is achieved by iterative exchange of messages between node k and its neighbors, using AC routines. A similar principle was used in [3] in order to develop a distributed implementation of the Oja-Karhunen recursion for eigenvectors, as discussed also in Section I. Once the elements $v_{(j)}[k]$ are available at the K nodes, inner product and norms necessary for eigenvalue computation are performed again by AC, while all element-wise operations (sums, multiplications by constants) are done locally at each node. Next, for both PM and LA, we first rewrite the global iteration in a way that is amenable to decentralized computation via AC, and then we break the global iteration down into a sequence of algorithmic steps to be executed by individual nodes.

A. Decentralized Power Method

The main result for the PM vector iteration is given by the following proposition.

Proposition 1: Given an ideal AC routine $\mathtt{AC}_{\cdot}(\cdot)$, the PM iteration (7) can be rewritten as

$$\mathbf{v}_{(j+1)} = \frac{K}{N} \mathrm{diag}\big\{ \mathbf{Y} \cdot \big[\mathtt{AC}_N\big(\mathrm{Diag}(\mathbf{v}_{(j)}^*) \cdot \mathbf{Y} \big) \big]^H \big\}. \tag{17}$$

Proof: From (7) we can write

$$\mathbf{v}_{(j+1)} = \frac{1}{N}\mathbf{Y}\mathbf{Y}^H\mathbf{v}_{(j)} \tag{18}$$

$$= \frac{1}{N}\mathrm{diag}(\mathbf{Y}\mathbf{Y}^H\mathbf{v}_{(j)}\mathbf{1}_K^T) \tag{19}$$

$$= \frac{1}{N}\mathrm{diag}\big(\mathbf{Y}\mathbf{Y}^H\mathrm{Diag}(\mathbf{v}_{(j)})\mathbf{1}\mathbf{1}^T\big) \tag{20}$$

$$= \frac{1}{N}\mathrm{diag}\big[\mathbf{Y}\big(\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(j)}^*)\mathbf{Y}\big)^H\big]. \tag{21}$$

Now, if we let $\mathbf{Z} \triangleq \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(j)}^*)\mathbf{Y}$ and $\mathbf{Z}_0 \triangleq \mathrm{Diag}(\mathbf{v}_{(j)}^*)\mathbf{Y}$, it is clear from (4) that $\mathbf{Z}$ is the ideal AC output with $\mathbf{Z}_0$ as initial matrix:

$$\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(j)}^*)\mathbf{Y} = K \cdot \mathtt{AC}_N\big(\mathrm{Diag}(\mathbf{v}_{(j)}^*)\mathbf{Y}\big). \tag{22}$$

Combining (21) with (22) yields (17). □

Complicated though it may look, expression (17) naturally leads to a decentralized implementation thanks to the following properties: *(i)* the k-th element of the output vector, $v_{(j+1)}[k]$, is the k-th element of $\mathrm{diag}(\mathbf{Y}\mathbf{Z}^H)$, hence

$$v_{(j+1)}[k] = \mathbf{z}[k]^H\mathbf{y}[k] \tag{23}$$

which can be computed locally by node k (recall that $\mathbf{y}[k]$ is the vector of samples received by node k and $\mathbf{z}[k]$ is the N-dimensional AC output at node k); *(ii)* the input to $\mathtt{AC}_N(\cdot)$ is such that the k-th row only contains node k's local data $v_{(j)}[k]^* \cdot \mathbf{y}[k]^T$ (recall that $v_{(j)}[k]$ has been computed by node k in the preceding iteration).

Assume now that the DPM iteration (17) has been repeated M times.[3] The largest eigenvalue estimate $\hat{\lambda}_1$ (8) can be computed for all nodes by two additional calls to AC, as follows from the following proposition.

Proposition 2: Given an ideal AC routine $\mathtt{AC}.(\cdot)$, after M iterations of (17), K local copies of the largest eigenvalue estimate (8) can be obtained as

$$\hat{\lambda}_1\mathbf{1}_K = \frac{K}{N}\mathtt{RN}\big[\mathtt{AC}_N\big(\mathrm{Diag}(\mathbf{v}_{(M)}^*)\cdot\mathbf{Y}\big)\big] \oslash \mathtt{AC}_1(\mathbf{v}_{(M)}^* \odot \mathbf{v}_{(M)}). \tag{24}$$

where $\mathtt{RN} : \mathbb{C}^{K\times m} \to \mathbb{C}^K$ is a function that returns the squared ℓ^2 norms of the rows of an input matrix (see Tab. I).

[3]Note that the number of algorithm iterations is denoted by M because M is in fact the dimension of the underlying Krylov subspace. This identity is more evident in the case of the LA.

Proof: We first write the global output for all K nodes after M iterations as

$$\hat{\lambda}_1 \mathbf{1}_K = \big(\mathbf{1}_K \mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)}\big) \oslash \big(\mathbf{1}_K \mathbf{v}_{(M)}^H \mathbf{v}_{(M)}\big), \tag{25}$$

where we have used (8) and applied element-wise division. The numerator can be written as

$$\begin{aligned}
\mathbf{1}\mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)} &= \frac{1}{N}\mathbf{1}\mathbf{v}_{(M)}^H \mathbf{Y}\mathbf{Y}^H \mathbf{v}_{(M)} && (26)\\
&= \frac{1}{N}\mathbf{1}\big\|\mathbf{v}_{(M)}^H \mathbf{Y}\big\|^2 && (27)\\
&= \frac{1}{N}\mathbf{1}\big\|\mathbf{1}^T \mathrm{Diag}(\mathbf{v}_{(M)}^*)\mathbf{Y}\big\|^2 && (28)\\
&= \frac{1}{N}\mathtt{RN}\big[\mathbf{1}\mathbf{1}^T \mathrm{Diag}(\mathbf{v}_{(M)}^*)\mathbf{Y}\big] && (29)\\
&= \frac{K^2}{N}\mathtt{RN}\big[\mathtt{AC}_N\big(\mathrm{Diag}(\mathbf{v}_{(M)}^*)\mathbf{Y}\big)\big]. && (30)
\end{aligned}$$

For the denominator, we can write

$$\begin{aligned}
\mathbf{1}\mathbf{v}_{(M)}^H \mathbf{v}_{(M)} &= \mathbf{1}\mathbf{1}^T(\mathbf{v}_{(M)}^* \odot \mathbf{v}_{(M)}) && (31)\\
&= K \cdot \mathtt{AC}_1(\mathbf{v}_{(M)}^* \odot \mathbf{v}_{(M)}). && (32)
\end{aligned}$$

Combining (30) with (32) and simplifying K finally yields (24). □

Similar to (17), expression (24) can be readily implemented in a distributed manner. At the numerator, every node k computes an AC vector $\mathbf{z}[k] \in \mathbb{C}^N$ starting from the initial value $v_{(M)}[k]^* \cdot \mathbf{y}[k]^T$, just like in the vector iteration, and takes the norm of $\mathbf{z}[k]$ locally. At the denominator, the scalar AC input at node k is simply the local quantity $|v_{(M)}[k]|^2$. Element-wise division is then performed internally by each node.

Thanks to the results of Propositions 1 and 2, and replacing the ideal AC routine by one with finite averaging time t, the DPM can be written in algorithmic form as summarized in Alg. 1. The averaging time or number of iterations t is assumed to be either predefined or adjusted online at every iteration, based on the starting vector and a target error bound.

B. Decentralized Lanczos Algorithm

Consider now the LA iteration (12)-(15). We note that (12) has the same structure as the numerator of (8), and (13) is similar to the PM iteration (7), with additional terms which can be computed locally provided that each node k has stored a local copy of $\alpha_{(j)}$, $\beta_{(j)}$, and of the k-th element of vectors $\mathbf{v}_{(j)}$, $\mathbf{v}_{(j-1)}$. Then, the normalization step (14)-(15) is similar to the

Algorithm 1: Decentralized power method

Input : Received signal vectors $\mathbf{y}[k] \in \mathbb{C}^N$ for $k \in \{1, \ldots, K\}$; number of iterations M; starting values $v_{(0)}[k]\ \forall k$; averaging time t.

Output: Eigenvalue estimates $\hat{\lambda}_1[k]\ \forall k$.

for *all nodes* k *in parallel* **do**
  for *iteration* $j = 1$ *to* M **do**
    $\mathbf{z}[k]^T = \mathtt{AC}_N^t[k]\left(v_{(j-1)}[k]^* \cdot \mathbf{y}[k]^T\right)$;
    Compute locally $v_{(j)}[k] = \frac{K}{N}\mathbf{z}[k]^H\mathbf{y}[k]$;
  end
  $\mathbf{z}[k]^T = \mathtt{AC}_N^t[k]\left(v_{(M)}[k]^* \cdot \mathbf{y}[k]^T\right)$;
  $d[k] = \mathtt{AC}_1^t[k]\left(|v_{(M)}[k]|^2\right)$;
  Compute locally $\hat{\lambda}_1[k] = \frac{K}{N} \cdot \|\mathbf{z}[k]\|^2/d[k]$;
end

denominator of (8), therefore it can be implemented by scalar AC and element-wise division. Based on the above observations, we can state the following result.

Proposition 3: Given an ideal AC routine $\mathtt{AC}_\cdot(\cdot)$, the LA (12)-(15) can be rewritten as

$$\alpha_{(j)}\mathbf{1}_K = \frac{K^2}{N}\mathtt{RN}\left[\mathtt{AC}_N\left(\mathrm{Diag}(\mathbf{v}_{(j)}^*)\cdot\mathbf{Y}\right)\right] \tag{33}$$

$$\mathbf{w}_{(j)} = \frac{K}{N}\mathrm{diag}\left\{\mathbf{Y}\cdot\left[\mathtt{AC}_N\left(\mathrm{Diag}(\mathbf{v}_{(j)}^*)\cdot\mathbf{Y}\right)\right]^H\right\} - \alpha_{(j)}\mathbf{v}_{(j)} - \beta_{(j)}\mathbf{v}_{(j-1)} \tag{34}$$

$$\beta_{(j+1)}^2\mathbf{1}_K = K\cdot\mathtt{AC}_1\left(\mathbf{w}_{(j)}^*\odot\mathbf{w}_{(j)}\right) \tag{35}$$

$$\mathbf{v}_{(j+1)} = \mathbf{w}_{(j)}/\beta_{(j+1)}. \tag{36}$$

Proof: The proof is a combination of the same steps already used in Prop. 1 and Prop. 2. More precisely, (33) follows from (12) using (26)-(30); (34) from (13) using (17); (35) from (14) using (31)-(32); and (36) is simply (15). □

The above result can be mapped to the decentralized algorithm reported in Alg. 2, where a realistic AC scheme $\mathtt{AC}_\cdot^t$ is adopted.

Algorithm 2: Decentralized Lanczos algorithm

Input : Received signal vectors $\mathbf{y}[k] \in \mathbb{C}^N$ for $k \in \{1, \ldots, K\}$; number of iterations $M \leq K$; starting values $v_{(1)}[k]\ \forall k$, such that $\sum_{k=1}^{K} |v_{(1)}[k]|^2 = 1$, $\beta_{(1)}[k] = 0\ \forall k$; averaging time t.

Output: Eigenvalue estimates $\{\hat{\lambda}_1[k], \ldots, \hat{\lambda}_M[k]\}\ \forall k$.

1 **for** *all nodes* k *in parallel* **do**
2 **for** *iteration* $j = 1$ *to* M **do**
3 $\mathbf{z}[k]^T = \mathtt{AC}_N^t[k]\left(v_{(j)}[k]^* \cdot \mathbf{y}[k]^T\right)$;
4 Compute locally $\alpha_{(j)}[k] = \frac{K^2}{N}\|\mathbf{z}[k]\|^2$;
5 Compute locally $w_{(j)}[k] = \frac{K}{N}\mathbf{z}[k]^H\mathbf{y}[k] - \alpha_{(j)}[k] \cdot v_{(j)}[k] - \beta_{(j)}[k] \cdot v_{(j-1)}[k]$;
6 $b[k] = \mathtt{AC}_1^t[k](|w_{(j)}[k]|^2)$;
7 Compute locally $\beta_{(j+1)}[k] = \sqrt{K \cdot b[k]}$ and $v_{(j+1)}[k] = w_{(j)}[k]/\beta_{(j+1)}[k]$;
8 **end**
9 Construct locally $\mathbf{T}[k]$ from (11) using $\alpha_{(1)}[k], \ldots, \alpha_{(M)}[k]$, $\beta_{(2)}[k], \ldots, \beta_{(M)}[k]$;
10 Compute locally $\{\hat{\lambda}_1[k], \ldots, \hat{\lambda}_M[k]\}$ = eigenvalues of $\mathbf{T}[k]$;
11 **end**

V. Error and Complexity Analysis

In this section we analyze the impact of non-ideal AC algorithms on the error and numerical complexity (especially in terms of network signaling) for the DPM and the DLA.

A. DPM Error

The DPM algorithm (Alg. 1) involves three sources of error due to AC. We define the first error term as

$$\mathbf{E}_{(j)}^{\mathsf{PM1}} \triangleq \mathtt{AC}_N^t\big(\mathrm{Diag}(\mathbf{v}_{(j-1)}^*) \cdot \mathbf{Y}\big) - \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(j-1)}^*) \cdot \mathbf{Y} \in \mathbb{C}^{K\times N}, \tag{37}$$

which occurs at every iteration of (17), corresponding to line 3 of Alg. 1. Following the previously used convention, we denote by $\mathbf{e}_{(j)}^{\mathsf{PM1}}[k]^T$ the k-th row of $\mathbf{E}_{(j)}^{\mathsf{PM1}}$. The second and third error terms

arise from (24), i.e., lines 6 and 7 of the algorithm, and are defined as

$$\mathbf{E}^{\mathsf{PM2}} \triangleq \mathtt{AC}_N^t\big(\mathrm{Diag}(\mathbf{v}_{(M)}^*)\cdot\mathbf{Y}\big) - \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(M)}^*)\cdot\mathbf{Y} \in \mathbb{C}^{K\times N}, \tag{38}$$

$$\mathbf{e}^{\mathsf{PM3}} \triangleq \mathtt{AC}_1^t(\mathbf{v}_{(M)}^* \odot \mathbf{v}_{(M)}) - \frac{1}{K}\mathbf{1}\mathbf{1}^T(\mathbf{v}_{(M)}^* \odot \mathbf{v}_{(M)}) \in \mathbb{C}^{K}. \tag{39}$$

Again, we refer to the k-th row of $\mathbf{E}^{\mathsf{PM2}}$ as $\mathbf{e}_{(j)}^{\mathsf{PM2}}[k]^T$ and to the k-th element of $\mathbf{e}^{\mathsf{PM3}}$ as $e^{\mathsf{PM3}}[k]$. Note that the three above defined errors are all instances of the general formula (4) applied with different inputs. To simplify the notation, we have dropped subscript t in the symbols of error variables.

With regard to the DPM convergence, the most important term is clearly $\mathbf{E}^{\mathsf{PM1}}$, because this error is added at each iteration of the algorithm. The impact of $\mathbf{E}^{\mathsf{PM1}}$ on the evolution of PM vectors $\mathbf{v}_{(j)}$ is expressed by the following result.

Proposition 4: Given a non-ideal AC scheme that introduces an error term $\mathbf{E}_{(j)}^{\mathsf{PM1}}$ as defined in (37), the resulting DPM vector after M iterations can be written as

$$\mathbf{v}_{(M)} = \mathbf{R}^M\mathbf{v}_{(0)} + \frac{1}{N}\sum_{j=1}^{M}\mathbf{R}^{M-j}\mathrm{diag}\big[\mathbf{Y}\big(\mathbf{E}_{(j)}^{\mathsf{PM1}}\big)^H\big]. \tag{40}$$

Proof: By applying the same steps as in the proof of Prop. 1 with the AC function defined in (4) with $\mathbf{E}_t$ replaced by $\mathbf{E}_{(j)}^{\mathsf{PM1}}$, we can write for any iteration j

$$\mathbf{v}_{(j)} = \mathbf{R}\mathbf{v}_{(j-1)} + \frac{1}{N}\mathrm{diag}\big[\mathbf{Y}\big(\mathbf{E}_{(j)}^{\mathsf{PM1}}\big)^H\big]. \tag{41}$$

The above formula, applied recursively for M iterations, yields (40). □

The term $\mathbf{R}^M\mathbf{v}_{(0)}$ in (40) represents the ideal PM output, while the summation on the r.h.s. represents the error. For brevity, we define

$$\mathbf{d}_{(j)} \triangleq \frac{1}{N}\mathrm{diag}\big[\mathbf{Y}\big(\mathbf{E}_{(j)}^{\mathsf{PM1}}\big)^H\big], \tag{42}$$

which represents the error introduced by AC in a single iteration j. Its k-th element (i.e., the component for node k) is $d_{(j)}[k] = \frac{1}{N}(\mathbf{e}_{(j)}[k])^H\mathbf{y}[k]$.

Now, the convergence of the DPM depends on the relative magnitude of the error term $\sum_{j=1}^{M}\mathbf{R}^{M-j}\mathbf{d}_{(j)}$ compared to the desired term $\mathbf{R}^M\mathbf{v}_{(0)}$ as $M\to\infty$ (recall that both terms are unnormalized). Let us assume that the spectral radius of $\mathbf{R}$ (i.e., λ_1, since the eigenvalues are real and positive) is larger than 1.[4] Then, the DPM error converges asymptotically to zero if

[4]This assumption simplifies the mathematical analysis, and it is not a limitation in practice, because it is always possible to rescale the data samples $\mathbf{Y}$ such that $\lambda_1 > 1$.

the magnitude of the AC error vector grows slower than a certain rate. The result is expressed formally by the following proposition.

Proposition 5: Let $\theta_{(j)} \in [0, \pi/2]$ be the angle between the true eigenvector and its DPM estimate $\mathbf{v}_{(j)}$ after j iterations, defined by

$$\cos\theta_{(j)} = \mathbf{u}_1^H \frac{\mathbf{v}_{(j)}}{\|\mathbf{v}_{(j)}\|}, \quad 0 \le j \le M, \tag{43}$$

and assume $\cos\theta_{(0)} \neq 0$, $\lambda_1 > 1$.
Then, asymptotically in M, provided that $\|\mathbf{d}_{(M)}\|_\infty = o\left(\frac{1}{M+1}\left(\frac{\lambda_1}{\max\{\lambda_2,1\}}\right)^M\right)$, we have

$$\lim_{M\to\infty} |\sin\theta_{(M)}| = 0. \tag{44}$$

Proof: Similarly as in [22, p. 406], we start by expressing vectors $\mathbf{v}_{(0)}$ and $\mathbf{d}_{(j)}$ in the eigenbasis $(\mathbf{u}_1, \ldots, \mathbf{u}_K)$, so that

$$\mathbf{v}_{(0)} = a_{0,1}\mathbf{u}_1 + \ldots + a_{0,K}\mathbf{u}_K, \tag{45}$$

$$\mathbf{d}_{(j)} = a_{j,1}\mathbf{u}_1 + \ldots + a_{j,K}\mathbf{u}_K, \quad 1 \le j \le M. \tag{46}$$

By assumption we have $|a_{0,1}| = \cos\theta_{(0)} \neq 0$. The DPM vector after M iterations can be then written as

$$\mathbf{v}_{(M)} = \mathbf{R}^M\mathbf{v}_{(0)} + \sum_{j=1}^{M}\mathbf{R}^{M-j}\mathbf{d}_{(j)} \tag{47}$$

$$= \sum_{j=0}^{M} a_{j,1}\lambda_1^{M-j}\mathbf{u}_1 + \ldots + \sum_{j=0}^{M} a_{j,K}\lambda_K^{M-j}\mathbf{u}_K, \tag{48}$$

and consequently

$$|\sin\theta_{(M)}|^2 = 1 - \frac{\left|\mathbf{u}_1^H\mathbf{v}_{(M)}\right|^2}{\|\mathbf{v}_{(M)}\|^2} \tag{49}$$

$$= 1 - \frac{\left|\sum_{j=0}^{M} a_{j,1}\lambda_1^{M-j}\right|^2}{\sum_{i=1}^{K}\left|\sum_{j=0}^{M} a_{j,i}\lambda_i^{M-j}\right|^2} \tag{50}$$

$$= \frac{\sum_{i=2}^{K}\left|\sum_{j=0}^{M} a_{j,i}\lambda_i^{M-j}\right|^2}{\sum_{i=1}^{K}\left|\sum_{j=0}^{M} a_{j,i}\lambda_i^{M-j}\right|^2} \tag{51}$$

$$\le \frac{\sum_{i=2}^{K}\left|\sum_{j=0}^{M} a_{j,i}\lambda_i^{M-j}\right|^2}{\left|\sum_{j=0}^{M} a_{j,1}\lambda_1^{M-j}\right|^2}. \tag{52}$$

By letting $a_i(M) \triangleq \max_{0\leq j\leq M} |a_{j,i}|$ and dividing numerator and denominator by λ_1^{2M}, we have

$$(52) \leq \frac{\sum_{i=2}^{K} \left| a_i(M) \sum_{j=0}^{M} \lambda_i^{M-j} \right|^2}{\left| \sum_{j=0}^{M} a_{j,1} \lambda_1^{M-j} \right|^2} \tag{53}$$

$$= \frac{\sum_{i=2}^{K} a_i^{2M} \left| \sum_{j=0}^{M} (\lambda_i/\lambda_1)^M \lambda_i^{-j} \right|^2}{\left| \sum_{j=0}^{M} a_{j,1} \lambda_1^{-j} \right|^2}. \tag{54}$$

Let us first consider the denominator, which can be written as $\left| \sum_{j=0}^{M} a_{j,1} \overline{\lambda}_1^j \right|^2$, with $\overline{\lambda}_1 \triangleq 1/\lambda_1 \in (0,1)$. We have to consider two cases. *(i)* Assume the series is absolutely convergent, so that $\lim_{M\to\infty} \sum_{j=0}^{M} |a_{j,1}| < \infty$. Moreover, since $a_{0,1} \neq 0$, we have $\lim_{M\to\infty} \sum_{j=0}^{M} |a_{j,1}| = \alpha$ for some $\alpha > 0$. Now note that $\sum_{j=0}^{M} a_{j,1} \overline{\lambda}_1^j$ is the Z-transform of the sequence $\{a_{j,1}\}_{j=0}^{\infty}$ at $\overline{\lambda}_1$. So, as $\overline{\lambda}_1 < 1$, the Z-transform exists, and therefore the series converges to a value β. *(ii)* Assume now that $\lim_{M\to\infty} \sum_{j=0}^{M} |a_{j,1}| = \infty$. In this case, the radius of convergence of the power series in $\overline{\lambda}_1$ is 0, which means that the series diverges for any value of $\overline{\lambda}_1$. Combining the two cases, we conclude that $\lim_{M\to\infty} \left| \sum_{j=0}^{M} a_{j,1} \lambda_1^{-j} \right|^2 \in [\beta^2, \infty]$, which means that in the worst case the denominator converges to a finite positive value.

Consider now the numerator, which we can write as

$$A_M \triangleq \sum_{i=2}^{K} a_i(M)^2 \underbrace{\left| \sum_{j=0}^{M} (\lambda_i/\lambda_1)^M \lambda_i^{-j} \right|^2}_{\triangleq A_M^{(i)}}. \tag{55}$$

Again, we have two cases. *(i)* If $\lambda_i \geq 1$, then $\lambda_i^j \geq 1$ for any $0 \leq j \leq M$, hence

$$A_M^{(i)} \leq \left| \sum_{j=0}^{M} (\lambda_i/\lambda_1)^M \right|^2 = \left| (M+1)(\lambda_i/\lambda_1)^M \right|^2, \tag{56}$$

and, recalling that $\lambda_i < \lambda_1$ for any $i \geq 2$ and applying L'Hopital's rule, we obtain $\lim_{M\to\infty} A_M^{(i)} = 0$. *(ii)* If $\lambda_i < 1$, we have

$$A_M^{(i)} = \left| \sum_{j=0}^{M} \lambda_1^{-M} \lambda_i^{M-j} \right|^2 \leq \left| \sum_{j=0}^{M} \lambda_1^{-M} \right|^2 = \left| (M+1)\lambda_1^{-M} \right|^2, \tag{57}$$

whose limit is again 0. Combining these results yields

$$A_M \leq \sum_{i=2}^{K} a_i(M)^2 \left| (M+1) \left(\frac{\max\{\lambda_2, 1\}}{\lambda_1} \right)^M \right|^2. \tag{58}$$

Now suppose that $\max_{2\leq i\leq K} a_i(M) \leq \gamma(M)$. Then, the numerator converges to 0 if $\gamma(M) = o\left(\frac{1}{M+1}\left(\frac{\lambda_1}{\max\{\lambda_2,1\}}\right)^M\right)$. The fact that $\max_{2\leq i\leq K} a_i(M)$ is equal (up to a constant) to $\|\mathbf{d}_{(M)}\|_\infty$ completes the proof. □

We next consider the impact of error terms $\mathbf{E}^{\mathsf{PM2}}$ and $\mathbf{e}^{\mathsf{PM3}}$, which only concern the eigenvalue computation phase at the final iteration. Let $e_{\mathsf{num}}[k]$ and $e_{\mathsf{den}}[k]$ be the errors introduced by non-ideal AC in step 8 of Alg. 1, defined such that

$$\hat{\lambda}_1[k] = \frac{\mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)} + e_{\mathsf{num}}[k]}{\mathbf{v}_{(M)}^H \mathbf{v}_{(M)} + e_{\mathsf{den}}[k]}. \tag{59}$$

The following proposition provides the values of $e_{\mathsf{den}}[k]$ and $e_{\mathsf{num}}[k]$ as a function of $\mathbf{E}^{\mathsf{PM2}}$ and $\mathbf{e}^{\mathsf{PM3}}$.

Proposition 6: Given a non-ideal AC scheme that introduces error terms $\mathbf{E}^{\mathsf{PM2}}$ and $\mathbf{e}^{\mathsf{PM3}}$ defined in (38) and (39), the resulting errors in $\hat{\lambda}_1[k]$ are given by

$$e_{\mathsf{num}}[k] = \frac{K}{N}\left[(\mathbf{e}^{\mathsf{PM2}}[k])^T \mathbf{Y}^H \mathbf{v}_{(M)} + \mathbf{v}_{(M)}^H \mathbf{Y}(\mathbf{e}^{\mathsf{PM2}}[k])^*\right] + \frac{K^2}{N}\left\|\mathbf{e}^{\mathsf{PM2}}[k]\right\|^2, \tag{60}$$

$$e_{\mathsf{den}}[k] = K \cdot e^{\mathsf{PM3}}[k]. \tag{61}$$

Proof: Using the results of Prop. 2 and introducing non-ideal AC, we can write

$$\hat{\lambda}_1[k] = \frac{K}{N} \cdot \frac{\left\|\frac{1}{K}\mathbf{v}_{(M)}^H \mathbf{Y} + (\mathbf{e}^{\mathsf{PM2}}[k])^T\right\|^2}{\frac{1}{K}\|\mathbf{v}_{(M)}\|^2 + e^{\mathsf{PM3}}[k]} \tag{62}$$

$$= \frac{\frac{1}{N}\left\|\mathbf{v}_{(M)}^H \mathbf{Y}\right\|^2 + \frac{K}{N}\left[(\mathbf{e}^{\mathsf{PM2}}[k])^T \mathbf{Y}^H \mathbf{v}_{(M)} + \mathbf{v}_{(M)}^H \mathbf{Y}(\mathbf{e}^{\mathsf{PM2}}[k])^*\right] + \frac{K^2}{N}\|\mathbf{e}^{\mathsf{PM2}}[k]\|^2}{\|\mathbf{v}_{(M)}\|^2 + K e^{\mathsf{PM3}}[k]}. \tag{63}$$

Since $\frac{1}{N}\left\|\mathbf{v}_{(M)}^H \mathbf{Y}\right\|^2 = \mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)}$, we note that (63) is equivalent to (59), with error expressions $e_{\mathsf{num}}[k]$ and $e_{\mathsf{den}}[k]$ given, respectively, by (60) and (61). □

By simple algebraic manipulations, and letting $\hat{\lambda}_1^{\mathsf{id}} \triangleq \left(\mathbf{v}_{(M)}^H \mathbf{R}\mathbf{v}_{(M)}\right)/\|\mathbf{v}_{(M)}\|^2$ be the estimate of λ_1 at the M-th DPM iteration without eigenvalue computation errors, we can write

$$\hat{\lambda}_1[k] = \frac{1}{1 + e_{\mathsf{den}}[k]/\|\mathbf{v}_{(M)}\|^2} \cdot \hat{\lambda}_1^{\mathsf{id}} + \frac{e_{\mathsf{num}}[k]}{\|\mathbf{v}_{(M)}\|^2 + e_{\mathsf{den}}[k]}. \tag{64}$$

The above expression shows immediately that the eigenvalue computation error becomes asymptotically negligible, as $\|\mathbf{v}_{(M)}\|^2 \to \infty$ as $M \to \infty$.

B. DLA Error

The DLA involves two error terms due to AC, in lines 3 and 6 of Alg. 2. The first error arises from in (33)-(34) and is defined as

$$\mathbf{E}_{(j)}^{\mathsf{LA1}} \triangleq \mathtt{AC}_N^t\big(\mathrm{Diag}(\mathbf{v}_{(j)}^*)\cdot\mathbf{Y}\big) - \frac{1}{K}\mathbf{1}\mathbf{1}^T\mathrm{Diag}(\mathbf{v}_{(j)}^*)\cdot\mathbf{Y} \in \mathbb{C}^{K\times N}. \tag{65}$$

The second error originates from (35) and is defined as

$$\mathbf{e}_{(j)}^{\mathsf{LA2}} \triangleq \mathtt{AC}_1^t\big(\mathbf{w}_{(j)}^*\odot\mathbf{w}_{(j)}\big) - \frac{1}{K}\mathbf{1}\mathbf{1}^T\big(\mathbf{w}_{(j)}^*\odot\mathbf{w}_{(j)}\big) \in \mathbb{C}^K. \tag{66}$$

As usual, the k-th row of $\mathbf{E}_{(j)}^{\mathsf{LA1}}$ is denoted by $\mathbf{e}_{(j)}^{\mathsf{LA1}}[k]^T$, and the k-th element of $\mathbf{e}_{(j)}^{\mathsf{LA2}}$ by $e_{(j)}^{\mathsf{LA2}}[k]$. Both error terms occur at every iteration j of the algorithm. We are now interested in the impact of such errors on $\mathbf{w}_{(j)}$. First of all, we notice that the k-th component of vector $\mathbf{w}_{(j)}$ in the ideal LA (13) can be written as

$$w_{(j)}^{\mathsf{id}}[k] \triangleq \frac{K}{N}\mathbf{y}[k]^T\mathbf{Y}^H\mathbf{v}_{(j)} - (\mathbf{v}_{(j)}^H\mathbf{R}\mathbf{v}_{(j)})\cdot v_{(j)}[k] - \|\mathbf{w}_{(j-1)}\|\cdot v_{(j-1)}[k], \tag{67}$$

by exploiting the expressions of $\alpha_{(j)}$ (12) and $\beta_{(j)}$ (14). We then define the error on $w_{(j)}[k]$ to be

$$e_{w_{(j)}}[k] \triangleq w_{(j)}[k] - w_{(j)}^{\mathsf{id}}[k]. \tag{68}$$

A closed-form expression of $e_{w(j)}[k]$ as a function of $\mathbf{E}_{(j)}^{\mathsf{LA1}}$ and $\mathbf{e}_{(j)}^{\mathsf{LA2}}$ on $e_{w(j)}[k]$ is given by the following Proposition. Later, we provide an interpretation of this error expression, and we discuss its impact on the estimation of eigenvalues.

Proposition 7: Given a non-ideal AC scheme that introduces error terms $\mathbf{E}_{(j)}^{\mathsf{LA1}}$ and $\mathbf{e}_{(j)}^{\mathsf{LA2}}$ defined respectively in (65) and (66), the resulting error in $w_{(j)}[k]$ is given by

$$\begin{aligned} e_{w(j)}[k] =& \frac{1}{N}(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^H\mathbf{y}[k] - \frac{K}{N}\left[(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^T\mathbf{Y}^H\mathbf{v}_{(M)} + \mathbf{v}_{(M)}^H\mathbf{Y}(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^*\right]\cdot v_{(j)}[k] \\ & - \frac{K^2}{N}\|\mathbf{e}_{(j)}^{\mathsf{LA1}}[k]\|^2\cdot v_{(j)}[k] - \frac{K}{2\|\mathbf{w}_{(j-1)}\|}e_{(j)}^{\mathsf{LA2}}[k]\cdot v_{(j-1)}[k] + o(e_{(j)}^{\mathsf{LA2}}[k]). \end{aligned} \tag{69}$$

Proof: The LA iteration (13), as well as the decentralized version (34), consists of three additive terms, therefore the error (68) can be expressed as the sum of three separate terms:

$$e_{w_{(j)}}[k] = e_{w_{(j)}}^{(\mathsf{I})}[k] + e_{w_{(j)}}^{(\mathsf{II})}[k] + e_{w_{(j)}}^{(\mathsf{III})}[k]. \tag{70}$$

The first term originates from the computation of $\frac{K}{N}\mathrm{diag}\big\{\mathbf{Y}\cdot\big[\mathtt{AC}_N\big(\mathrm{Diag}(\mathbf{v}_{(j)}^*)\cdot\mathbf{Y}\big)\big]^H\big\}$ in (34), which is identical to the PM iteration. Thus, the first error term can be expressed using (41)

with $\mathbf{E}_{(j)}^{\mathsf{PM1}}$ replaced by $\mathbf{E}_{(j)}^{\mathsf{LA1}}$. The global error for all nodes is $\frac{1}{N}\mathrm{diag}\big[\mathbf{Y}\big(\mathbf{E}_{(j)}^{\mathsf{LA1}}\big)^H\big]$, and its k-th element is

$$e_{w_{(j)}}^{(\mathsf{I})}[k] = \frac{1}{N}(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^H\mathbf{y}[k]. \tag{71}$$

The second term arises from computation of $\alpha_{(j)}[k]$, which is done using the same vector $\mathbf{z}[k]$ obtained via AC in line 3. Therefore the error on $\alpha_{(j)}[k]$ depends again on $\mathbf{E}_{(j)}^{\mathsf{LA1}}$. Since $\alpha_{(j)}[k]$ is calculated as the square norm of $\mathbf{z}[k]$ (line 4), the structure of the error is the same as that of $e_{\mathsf{num}}[k]$ for the DPM (60). By replacing $\mathbf{E}^{\mathsf{PM2}}$ by $\mathbf{E}_{(j)}^{\mathsf{LA1}}$ in (60) and multiplying by $v_{(j)}[k]$, we obtain the second part as

$$e_{w_{(j)}}^{(\mathsf{II})}[k] = \frac{K}{N}\left[(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^T\mathbf{Y}^H\mathbf{v}_{(M)} + \mathbf{v}_{(M)}^H\mathbf{Y}(\mathbf{e}_{(j)}^{\mathsf{LA1}}[k])^*\right]\cdot v_{(j)}[k] + \frac{K^2}{N}\|\mathbf{e}_{(j)}^{\mathsf{LA1}}[k]\|^2\cdot v_{(j)}[k]. \tag{72}$$

The third part contains the error due to computation of $\beta_{(j)}[k]$, i.e., the norm of $\mathbf{w}_{(j-1)}$. The error arises from the AC algorithm used in line 6 when computing $\|\mathbf{w}_{(j-1)}\|^2$; a nonlinearity is then introduced by the square root. Using a first-order Taylor expansion (under the assumption that $e_{(j)}^{\mathsf{LA2}}[k] \ll \|\mathbf{w}_{(j-1)}\|$), we can write

$$\begin{aligned}
\beta_{(j)}[k] &= \sqrt{\|\mathbf{w}_{(j-1)}\|^2 + Ke_{(j)}^{\mathsf{LA2}}[k]} && (73)\\
&= \|\mathbf{w}_{(j-1)}\|\sqrt{1 + \frac{K}{\|\mathbf{w}_{(j-1)}\|^2}e_{(j)}^{\mathsf{LA2}}[k]} && (74)\\
&= \|\mathbf{w}_{(j-1)}\|\left[1 + \frac{K}{2\|\mathbf{w}_{(j-1)}\|^2}e_{(j)}^{\mathsf{LA2}}[k] + o(e_{(j)}^{\mathsf{LA2}}[k])\right] && (75)\\
&= \|\mathbf{w}_{(j-1)}\| + \frac{K}{2\|\mathbf{w}_{(j-1)}\|}e_{(j)}^{\mathsf{LA2}}[k] + o(e_{(j)}^{\mathsf{LA2}}[k]), && (76)
\end{aligned}$$

hence the third error term is given by

$$e_{w_{(j)}}^{(\mathsf{III})}[k] = \frac{K}{2\|\mathbf{w}_{(j-1)}\|}e_{(j)}^{\mathsf{LA2}}[k]\cdot v_{(j-1)}[k] + o(e_{(j)}^{\mathsf{LA2}}[k]). \tag{77}$$

The resulting error (69) is obtained by summation of (71), (72), and (77). □

The error expressed by Prop. 7 is then propagated to the next iteration $j+1$ through another nonlinear step (line 7). Using the same procedure of Eqs. (73)-(76), the relation between $v_{(j+1)}[k]$ and $w_{(j)}[k]$ can be written as

$$v_{(j+1)}[k] = \frac{1}{1 + \frac{K}{\|\mathbf{w}_{(j-1)}\|^2}e_{(j+1)}^{\mathsf{LA2}}[k] + o(e_{(j+1)}^{\mathsf{LA2}}[k])}\cdot\frac{w_{(j)}[k]}{\|\mathbf{w}_{(j)}\|}. \tag{78}$$

In summary, (69) expresses the error on $\mathbf{w}_{(j)}$ given $\mathbf{v}_{(j)}$ and $\mathbf{v}_{(j-1)}$, and (78) expresses the error on $\mathbf{v}_{(j+1)}$ given $\mathbf{w}_{(j)}$. In principle, a closed-form error update rule could be derived from these

expressions (as it was done for the DPM), but the resulting formula would be too complicated to provide additional insight. We conclude the DLA analysis with two remarks.

1) Following the same line of reasoning of [26], the error term in (69) represents a *loss of orthogonality* of the vectors $\mathbf{v}_{(j)}$, while (78) results in a *loss of unit norm*. As documented in the literature, the loss of orthogonality leads to the appearance of so-called spurious eigenvalues at certain iterations (typically spurious eigenvalues are duplicates of existing eigenvalues already computed in the previous iterations). Some heuristics for the identification of spurious eigenvalues exist, such as the Cullum-Willoughby method [40], which compares the eigenvalues of $\mathbf{T}$ (11) with those of the same matrix without the first row and column. Alternatively, spurious values can be detected by comparing the eigenvalues of $\mathbf{T}$ at iterations j and $j-1$. In our application, another simple criterion can be derived by observing that the covariance matrix $\mathbf{R}$ has exactly $\min\{K, N\}$ non-zero eigenvalues, hence for iterations $j > \min\{K, N\}$ any new non-zero value is automatically identified as spurious. We remark that all criteria for detection of spurious eigenvalues are compatible with the DLA, as they involve local post-processing of the output at individual nodes (matrices $\mathbf{T}[k]$ as defined in line 9 of Alg. 2).

2) The terms $\frac{K}{2\|\mathbf{w}_{(j-1)}\|} e^{\mathsf{LA2}}_{(j)}[k] \cdot v_{(j-1)}[k]$ in (69) and $\frac{K}{\|\mathbf{w}_{(j-1)}\|^2} e^{\mathsf{LA2}}_{(j+1)}[k]$ in (78) may be critical for the convergence of the algorithm in the event that $\|\mathbf{w}_{(j)}\| \approx 0$ (or, equivalently, $\beta_{(j)} \approx 0$) at some iteration j. In the ideal LA, quoting [22], "a zero $\beta_{(j)}$ is a welcome event in that is signals the computation of an exact invariant subspace. However, an exact zero or even a small $\beta_{(j)}$ is a rarity in practice." This event is even more unlikely to happen in the case of the DLA: simulation results confirm that the values of $\beta_{(j)}$ are always far from zero. As a result, cases of divergence of the DLA were never observed. Nevertheless, the DLA turns out to be more sensitive to numerical problems than the DPM, as shown in Section VII.

C. Complexity

For both the DPM and the DLA, complexity mainly arises from the repeated use of AC routines, resulting in communication overhead, time delays, and possible synchronization issues. We next compare the complexity of the two algorithms in terms of the following parameters: *(i)* number of calls to functions $\mathtt{AC}_N$ and $\mathtt{AC}_1$; *(ii)* total number of "information units" exchanged over the wireless channel by one node with degree (number of neighbors) d, where an information unit is defined as the number of bits used to encode a scalar; *(iii)* number of algorithmic steps,

	Number of $\texttt{AC}_N$	Number of $\texttt{AC}_1$	Number of information units exchanged per node	Number of required time periods
DPM	$M+1$	1	$I(MN+N+1)d$	$M+2$
DLA	M	M	$I(MN+M)d$	$2M$

TABLE II

NUMERICAL COMPLEXITY OF DPM AND DLA. LEGEND: N = NUMBER OF SAMPLES, M = NUMBER OF DPM/DLA ITERATIONS, I = NUMBER OF AC ITERATIONS, d = NODE DEGREE.

defined as individual tasks that must be performed in a sequential way due to input-output dependency (i.e., step n cannot start until step $n-1$ is completed).

The above performance figures are reported in Table II for the two algorithms. For simplicity it is assumed that all instances of AC have the same number of iterations, I. It can be observed that the total number of calls to consensus routines is slightly higher for the DLA (assuming $M > 2$): the DPM requires one vector-AC per iteration (line 3) in addition to another vector-AC and a scalar-AC for eigenvalue computation (lines 6-7), whereas the DLA uses one vector-AC and one scalar-AC in each iteration (lines 3-6). The amount of information sent over the air is nearly the same, as shown in the third column of the table. However, the DLA is significantly more complex than the DPM in terms of the number of algorithmic steps: in the case of the DLA, each iteration consists of two sequential steps (vector computation and normalization) that cannot be parallelized, hence the total number of steps for the DLA is nearly twice that of the DPM.

The above results suggest that the values of M, N, and I should be kept as small as possible in order to reduce complexity and improve the reactivity of the algorithms, especially when used in real-time detection applications. In particular, the number of samples can be very small if the number of nodes K is large enough (large networks are the natural scenario of application for decentralized algorithms). For example, in a network of 40 nodes, 10 samples per node are sufficient to detect a signal with SNR of 7dB with high probability (see Fig. 5(b) in Section VII).

We now briefly analyze the computational complexity for individual nodes. For the DPM, the dominant factor is the vector product of line 4. Since the vector size is N and the product

Name	Test statistic	Application scenario	Ref.
Roy's test (RT)	$T_{\mathsf{R}} \triangleq \lambda_1/\sigma^2$	$P = 1$, known σ^2	[30], [32], [37]
GLR test (GT)	$T_{\mathsf{G}} \triangleq \lambda_1/\sum_{i=1}^{K} \lambda_i$	$P = 1$, unknown σ^2	[34]–[36]
Sphericity test (ST)	$T_{\mathsf{S}} \triangleq \prod_{i=1}^{K} \lambda_i / \left(\frac{1}{K}\sum_{i=1}^{K} \lambda_i\right)^K$	$P > 1$, finite SNR	[34], [35], [38]
John's test (JT)	$T_{\mathsf{J}} \triangleq \sum_{i=1}^{K} \lambda_i^2 / \left(\sum_{i=1}^{K} \lambda_i\right)^2$	$P > 1$, low SNR	[38]

TABLE III

EIGENVALUE-BASED TESTS FOR MULTI-SENSOR SIGNAL DETECTION.

is computed at every iteration, the computational complexity per node scales as $O(MN)$. In the DLA, every iteration involves computing the norm of a vector of size N (line 4) and a vector product of the same size (line 5), hence the computational complexity per node scales as $O(MN)$ as well. In addition, the DLA involves local calculation of the eigenvalues of the tridiagonal matrix $\mathbf{T}[k]$ (line 10), which has complexity $O(M^2)$. The dominant term between $O(MN)$ and $O(M^2)$ depends on the relative values of M and N.

VI. APPLICATION TO SPECTRUM SENSING

We consider a distributed spectrum sensing scenario, where multiple sensor nodes cooperate to detect the presence of a primary signal in a given frequency band. A decision about signal presence or absence is made upon receiving N signal samples at each sensor. The main challenge for cognitive networks is to achieve reliable signal detection with a limited number of samples. Mathematically, the problem is a binary hypothesis test between a "signal-plus-noise hypothesis" ($\mathcal{H}_1$) and a "noise-only" hypothesis ($\mathcal{H}_0$) based on the received samples $\mathbf{Y}$. The network is assumed to operate under homogeneous conditions, i.e., the same hypothesis ($\mathcal{H}_0$ or $\mathcal{H}_1$) holds for all sensors during the entire sensing period. Given the model already introduced in Section II and assuming zero-mean complex Gaussian noise, the signal vector received at a given time instant n at the K sensors can be written under $\mathcal{H}_0$ as

$$\mathbf{y}(n)|_{\mathcal{H}_0} = \boldsymbol{\eta}(n), \tag{79}$$

where $\boldsymbol{\eta}(n) \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \sigma^2\mathbf{I})$. Under $\mathcal{H}_1$, the received vector is

$$\mathbf{y}(n)|_{\mathcal{H}_1} = \mathbf{H}\mathbf{s}(n) + \boldsymbol{\eta}(n), \tag{80}$$

where $\mathbf{s}(n) \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, \mathrm{Diag}([\sigma_1^2, \ldots, \sigma_P^2])) \in \mathbb{C}^P$ is a vector of signal samples transmitted by P sources (primary users), modeled as zero-mean Gaussian mutually uncorrelated random variables, and $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_P] \in \mathbb{C}^{K \times P}$ is a complex channel matrix whose columns represent the channels between the P signal sources and the K sensors. The channel coefficients are assumed to be unknown but constant during the sensing period[5]. The SNR for the i-th signal source is defined (under $\mathcal{H}_1$) as $\rho_i \triangleq \|\mathbf{h}_i\|^2 \sigma_i^2 / \sigma^2$. Let $T(\mathbf{Y})$ be a generic test statistic for signal detection, computed from the signal samples, and let ϑ be the associated decision threshold, such that the detector decides for $\mathcal{H}_1$ if $T(\mathbf{Y}) > \vartheta$ and for $\mathcal{H}_0$ otherwise. Then, false-alarm and detection probabilities are defined, respectively, as $\mathsf{P_{fa}} \triangleq \Pr[T(\mathbf{Y}) > \vartheta | \mathcal{H}_0]$ and $\mathsf{P_d} \triangleq \Pr[T(\mathbf{Y}) > \vartheta | \mathcal{H}_1]$. Detectors are typically calibrated so as to achieve a fixed false-alarm rate $\mathsf{P_{fa}} = \alpha$, i.e., the threshold is set as $\vartheta(\alpha)$.

Signal detection in the above-described setting has been extensively studied in the cognitive radio literature, e.g., [30]–[38], leading to the derivation of several possible test statistics $T(\mathbf{Y})$. Most of such statistics are functions of the eigenvalues of the sample covariance matrix (some examples are reported in Table III) and, as such, they can be computed in a decentralized network by applying the DPM or the DLA proposed in this paper. The problem reduces to computing, for each sensor k, a local version of the adopted test statistic, say $\hat{T}_i[k]$ with $i \in \{\mathsf{R}, \mathsf{G}, \mathsf{S}, \mathsf{J}\}$. This is obtained directly as a function of the local eigenvalue estimates $\hat{\lambda}_j[k]$ (with $j = 1$ for the DPM, which is sufficient to compute the RT statistic; and $1 \leq j \leq K$ for the DLA). Then, each node tests the local statistic $\hat{T}_i[k]$ against the predefined threshold $\vartheta(\alpha)$ and decides for $\mathcal{H}_0$ or $\mathcal{H}_1$ according to the rule

$$\hat{T}_i[k] \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \vartheta(\alpha). \tag{81}$$

In order to average out the numerical errors introduced by non-ideal AC at different nodes[6], a final round of AC can be executed taking as inputs the local statistics $\hat{T}_i[k]$. The result

$$\hat{T}'_i[k] = \mathtt{AC}_1^t[k]\big(\hat{T}_i[k]\big) \tag{82}$$

shall be used in (81) instead of $\hat{T}_i[k]$.

[5]Assuming the sensing period to be shorter than the coherence time of the channel is reasonable in multi-sensor spectrum sensing applications, where accurate detection is achieved already at low sample size.

[6]Numerical errors may result in the problem of different decisions at different nodes if $\hat{T}_i[k]$ is very close to $\vartheta(\alpha)$.

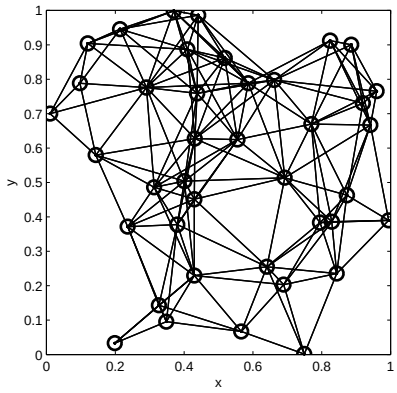


Fig. 1. Network topology.

The popular cooperative energy detector, i.e., the test based on the (possibly weighted) sum of the received signal energies at different sensors, also admits a natural decentralized implementation via AC algorithms. This problem was investigated in [39]. Decentralized energy detection is computationally simpler than eigenvalue-based techniques, but clearly inherits the well-known shortcomings of energy detection (suboptimality in multi-sensor settings and sensitivity to noise uncertainty).

In some applications, the goal is not only to discriminate between $\mathcal{H}_0$ and $\mathcal{H}_1$, but to estimate the number of signals P. This problem can be addressed again by eigenvalue-based estimators, such as the well-known Wax and Kailath's information-theoretic criteria [41], or the recent random matrix theory (RMT) estimator proposed in [42]. In all cases, the estimated number of signals is estimated as

$$\hat{P} = \arg \min_{0 \leq q \leq K} T(q; \lambda_1, \ldots, \lambda_K), \tag{83}$$

where $T(\cdot)$ is a function of the eigenvalues and, once again, can be computed locally from the DLA estimates $\hat{\lambda}_1, \ldots, \hat{\lambda}_K$.

VII. Simulation Results

For the purpose of numerical evaluation of the proposed algorithms, we consider a randomly generated network consisting of $K = 40$ nodes, as depicted in Fig. 1. Edges in the graph represent communication links between pairs of nodes. The received signal samples $\mathbf{Y}$ are randomly generated at every Monte Carlo iteration according to the signal-plus-noise model described in

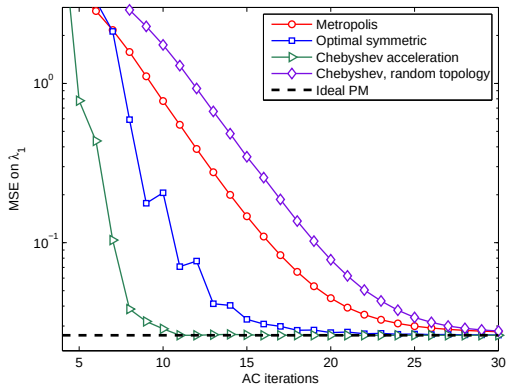


Fig. 2. Comparison of different AC algorithms.

Section VI (case $\mathcal{H}_1$), with the following parameters: $N = 10$ samples per node, noise variance $\sigma^2 = 1$, one Gaussian signal source ($P = 1$) with SNR $\rho = 5$dB.

We first turn our attention to the convergence of the largest eigenvalue, expressed in terms of the mean square error (MSE) of $\hat{\lambda}_1$ with respect to λ_1. In practice, the MSE is calculated as an average over 3000 Monte Carlo simulations. In Fig. 2 we compare the performance of different AC algorithms when applied in the DPM (Alg. 1). The number of DPM iterations is fixed to $M = 20$, while the number of AC iterations (I) varies from 5 to 30 (x-axis in the plot). We consider four alternative AC schemes, namely two versions of traditional AC [8] – with Metropolis weights (a heuristic based on the graph Laplacian) and with optimal symmetric weights (resulting from the solution of a convex optimization problem) – and two versions of AC with Chebyshev acceleration [19] – assuming first a fixed network topology and then one with 3% link failure probability. The lowest achievable bound is given by the ideal PM (i.e., a DPM with exact AC). The Chebyshev acceleration algorithm turns out to significantly outperform traditional AC even with optimized weights (under deterministic settings). For this reason we adopt Chebyshev-accelerated AC as the standard consensus scheme in all the following simulations.

In Fig. 3 we compare the DPM against the DLA. We observe that the DLA exhibits a faster convergence rate, but is more sensitive than the DPM to numerical errors introduced by imperfect consensus: the DLA error converges to 0 for $I = 15$ AC iterations, but not for $I = 10$, whereas the performance of the DPM is practically identical to that of the ideal PM already for $I = 10$

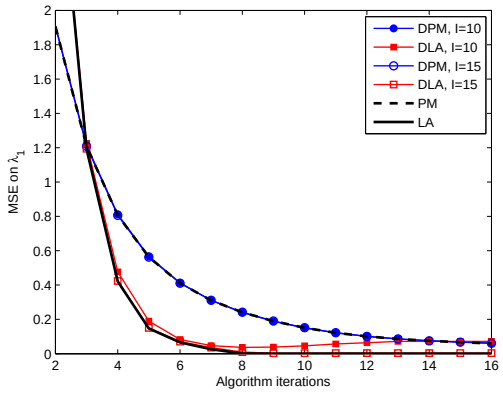
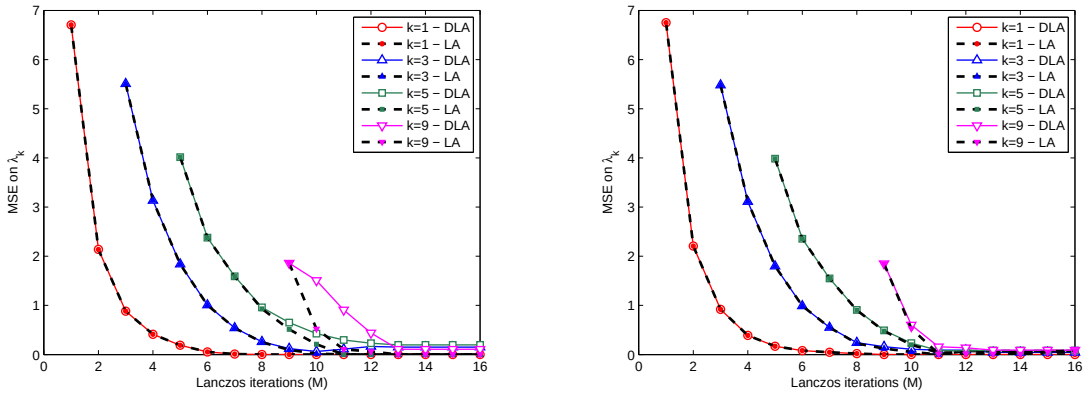


Fig. 3. Convergence of λ_1.

(and even for smaller values). In other words, the performance of the DLA compared to the DPM is better in absolute terms (especially with few algorithm iterations), but worse in relative terms. This behavior is consistent with the analysis presented in Section V and can be understood intuitively by noticing that each step of the DLA uses AC twice, in contrast to the DPM where AC is used only once per iteration.

We next evaluate the performance of the DLA when estimating multiple eigenvalues. The results, shown in Fig. 4, indicate that the higher is the order of eigenvalues to be estimated, the more iterations are needed. The reason is that, by definition of the LA, the j-th eigenvalue cannot be estimated until the j-th iteration; in addition, numerical errors sometimes cause the appearance of spurious eigenvalues (see Section V-B) which, even when correctly identified, introduce a one-step delay in the algorithm. One way to mitigate the numerical problems that affect the estimation of high-order eigenvalues is to improve the precision of the AC routine by tuning the parameter I. For example, according to our simulations, 20 AC iterations are sufficient to achieve an accurate estimation of the λ_1 and λ_3 (Fig. 4(a)), whereas 30 iterations are necessary for λ_5 and λ_9 (Fig. 4(b)).

We now illustrate an application of the proposed DPM and DLA for spectrum sensing in a distributed cognitive radio network. We assume again the same network topology of Fig. 1, with $K = 40$ and $N = 10$. According to the model introduced in Section VI, the samples under $\mathcal{H}_0$ are Gaussian distributed with variance $\sigma^2 = 1$, while under $\mathcal{H}_1$ we have one Gaussian signal component ($P = 1$) with SNR $\rho = 7$dB. We test the performance of two signal detectors: the RT



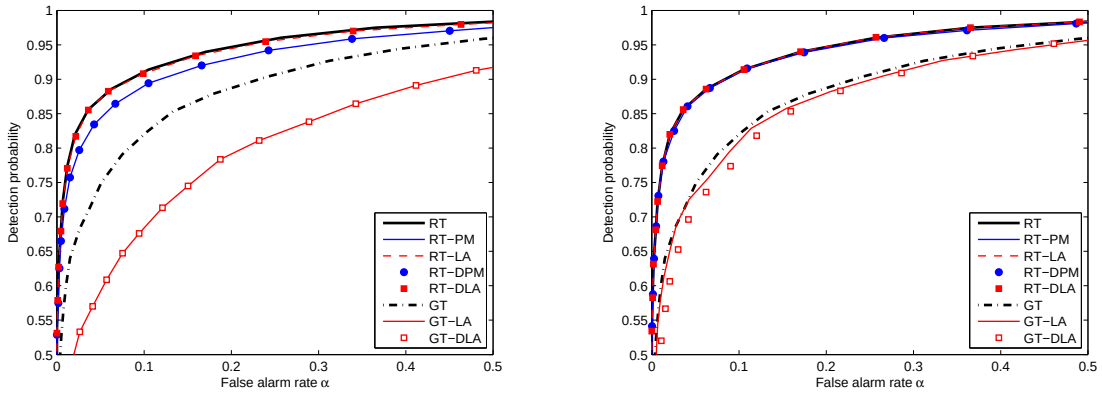
(a) $I = 20$ (b) $I = 30$

Fig. 4. Convergence of multiple eigenvalues.

and the GT, defined respectively as T_{R} and T_{G} in Tab. III. The RT is a test of significance of the largest eigenvalue (λ_1) alone, hence it can be implemented in a decentralized setting using either the DPM or the DLA. The GT, in contrast, involves all eigenvalues and requires the use of the DLA. In Fig. 5 we compare: *(i)* the ideal performance of RT and GT using the eigenvalues of $\mathbf{R}$ computed exactly by a fusion center with perfect communication links; *(ii)* the performance achieved when the eigenvalues are computed by the PM or LA (assuming ideal AC), respectively for $M = 5$ and $M = 10$; and *(iii)* the performance achieved when the eigenvalues are computed by the DPM or DLA, with $I = 30$ iterations. The results show that, after $M = 5$ algorithm iterations (Fig. 5(a)), the RT using LA or DLA already attains nearly-ideal performance, while the RT with DPM is slightly suboptimal; for the GT, on the other hand, 5 iterations are not enough to reach convergence of all eigenvalues (note that the performance gap is due to the LA itself and not to the decentralized version). After $M = 10$ iterations (Fig. 5(b)), all versions of the RT converge to the ideal RT bound, and the gap of LA- and DLA-GT compared to the ideal GT is dramatically reduced.

VIII. Conclusion

In this paper we have proposed and analyzed two general-purpose algorithms that can be used for computing sample covariance eigenvalues in distributed wireless networks. As an application, we have considered spectrum sensing in cognitive networks, and we have shown that numerous



(a) $M = 5$ (b) $M = 10$

Fig. 5. Signal detection: ROC curves.

eigenvalue-based tests for single-signal and multiple-signal detection can be implemented in a decentralized setting by using the proposed algorithms. Such decentralized signal detection techniques enable sensor nodes to compute global test statistics locally, thereby performing hypothesis tests without relying on a fusion center. Decentralized approaches also provide additional robustness to node failures or Byzantine attacks.

The two proposed algorithms – the DPM and the DLA – have different strengths and drawbacks. The DPM is less complex, more robust to numerical problems, and provably convergent even in the presence of non-ideal AC (under the conditions of Proposition 5); however, it can estimate only the largest eigenvalue, and under ideal assumptions its convergence rate is slower than that of the DLA. On the other hand, the DLA is able to estimate all eigenvalues (albeit with increasing complexity with the number of eigenvalues) and, even in the presence of AC errors, provides faster initial convergence than the DPM; however, it is more complex and more sensitive to numerical errors, and requires some post-processing in order to remove "spurious" eigenvalues.

Evaluated through simulations, both algorithms exhibit good performance in practical conditions (non-ideal AC algorithms, small number of samples) already after few iterations. In particular, convergence is very fast in the case of the largest eigenvalue, which results in high-performing distributed signal detectors based on the largest eigenvalue (referred to as RT-DPM and RT-DLA in Fig. 5). Other possible fields of applications of the proposed algorithms

are distributed anomaly detection in wireless sensor networks and signal feature estimation in distributed antenna arrays.

ACKNOWLEDGMENTS

The authors are grateful to colleagues Jafar Mohammadi and Meng Zheng for their valuable comments and discussions on some of the topics of this paper, and to Renato L. G. Cavalcante for providing the simulation code from [19].

REFERENCES

[1] F. Penna, S. Stanczak, "Decentralized Largest Eigenvalue Test for Multi-Sensor Signal Detection," *Proc. IEEE Global Communications Conference (Globecom)*, Anaheim, CA, USA, Dec. 2012.

[2] F. Penna, S. Stanczak, "Eigenvalue-based signal detection in cognitive femtocell networks using a decentralized Lanczos algorithm," *Proc. IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN) – Poster Session*, Bellevue, WA, USA, Oct. 2012.

[3] L. Li, A. Scaglione, J. H. Manton, "Distributed Principal Subspace Estimation in Wireless Sensor Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, Aug. 2011.

[4] P. Belanovic, S. Valcarcel Macua, S. Zazo, "Distributed static linear Gaussian models using consensus," *ELSEVIER Neural Networks*, no. 34, Oct. 2012.

[5] A. Wiesel, A. O. Hero, "Decomposable Principal Component Analysis", *IEEE Transactions on Signal Processing*, vol. 57, no. 11, Nov. 2009.

[6] S. B. Korada, A. Montanari, S. Oh, "Gossip PCA", *Proc. ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, San Jose, CA, USA, June 2011.

[7] R. Olfati-Saber, J. A. Fax, R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, no. 1, Jan. 2007.

[8] L. Xiao, S. Boyd, "Fast linear iterations for distributed averaging," *Elsevier Systems and Control Letters*, vol. 53, Feb. 2004.

[9] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed Average Consensus with Least-Mean-Square Deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1. Jan. 2007.

[10] R. Olfati-Saber, R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, Sept. 2004.

[11] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, "Randomized gossip algorithms,´´, *IEEE Transactions on Information Theory*, vol. 52, no. 6, June 2006.

[12] C. C. Moallemi, B. Van Roy, "Consensus Propagation," *IEEE Transactions on Information Theory*, vol. 52, no. 11, Nov. 2006.

[13] G. Scutari, S. Barbarossa, "Distributed Consensus Over Wireless Sensor Networks Affected by Multipath Fading," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, Aug. 2008.

[14] A. Tahbaz-Salehi, A. Jadbabaie, "Necessary and sufficient conditions for consensus over random networks," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, April 2008.

[15] R. Carli, G. Como, P. Frasca, F. Garin, "Average consensus on digital noisy networks," *Proc. 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys'09)*, Venice, Italy, Sept. 2009.

[16] B. Touri, A. Nedic, "Distributed consensus over network with noisy links," *Proc. 12th International Conference on Information Fusion*, July 2009.

[17] S. Kar, J. M. F. Moura, "Distributed Consensus Algorithms in Sensor Networks: Quantized Data and Random Link Failures," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, March 2010.

[18] R. L. G. Cavalcante, B. Mulgrew, "Adaptive Filter Algorithms for Accelerated Discrete-Time Consensus," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, March 2010.

[19] R. L. G. Cavalcante, A. Rogers, N. R. Jennings, "Consensus acceleration in multiagent systems with the Chebyshev semi-iterative method", *Proc. 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Taipei, Taiwan, May 2011.

[20] M. Zheng, M. Goldenbaum, S. Stanczak, H. Yu, "Fast average consensus in clustered wireless sensor networks by superposition gossiping,", *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Paris, France, Apr. 2012.

[21] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," *Proc. Amer. Control Conf. (ACC)*, New York, NY, USA, July 2007.

[22] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, MD, 1989.

[23] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,", *J. Res. Nat. Bur. Standards*, vol. 45, pp. 255–282, 1950.

[24] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003.

[25] C. C. Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," *J. Inst. Maths Applics*, vol. 10, pp. 373–381, 1972.

[26] C. C. Paige, "Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix,", *J. Inst. Maths Applics*, vol. 18, pp. 341–349, 1976.

[27] C. C. Paige, "Accuracy and Effectiveness of the Lanczos Algorithm for the Symmetric Eigenproblem,", *ELSEVIER Linear Algebra and its Applications*, vol. 34, pp. 235–258, Dec. 1980.

[28] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, 2000.

[29] A. Pothen, D. H. Simon, K. P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal of Matrix Analysis and Applications*, vol. 11, pp. 430-452, 1990.

[30] Y. Zeng, Y.-C. Liang, R. Zhang, "Blindly combined energy detection for spectrum sensing in cognitive radio," *IEEE Signal Processing Letters*, vol. 15, 2008.

[31] F. Penna, R. Garello, M. A. Spirito, "Cooperative Spectrum Sensing based on the Limiting Eigenvalue Ratio Distribution in Wishart Matrices," *IEEE Comm. Letters*, vol. 13, no. 7, July 2009.

[32] L. Wei, O. Tirkkonen, "Cooperative Spectrum Sensing of OFDM Signals Using Largest Eigenvalue Distributions," *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Tokyo, Japan, Sept. 2009.

[33] Y. H. Zeng and Y.-C. Liang, "Eigenvalue based spectrum sensing algorithms for cognitive radio," *IEEE Trans. on Communications*, vol. 57, no. 6, June 2010.

[34] R. Zhang, T.J. Lim, Y.C. Liang, Y. Zeng, "Multi-antenna based spectrum sensing for cognitive radios: a GLRT approach", *IEEE Trans. on Communications*, vol. 58, no. 1, Jan. 2010.

[35] A. Taherpour, M. Nasiri-Kenari, S. Gazor, "Multiple Antenna Spectrum Sensing in Cognitive Radios," *IEEE Trans. on Wireless Communications*, vol. 9, no. 2, Feb. 2010.

[36] P. Bianchi, M. Debbah, M. Maida, J. Najim, "Performance of Statistical Tests for Single-Source Detection Using Random Matrix Theory," *IEEE Transactions on Information Theory*, vol. 57, no. 4, Apr. 2011.

[37] B. Nadler, F. Penna, R. Garello, "Performance of Eigenvalue-based Signal Detectors with Known and Unknown Noise Level," *Proc. IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 2011.

[38] L. We, O. Tirkkonen, "Spectrum Sensing in the Presence of Multiple Primary Users,´´ *IEEE Transactions on Communications*, vol. 60, no. 5, May 2012.

[39] Z. Li, F. R. Yu, M. Huang, "A Distributed Consensus-Based Cooperative Spectrum-Sensing Scheme in Cognitive Radios," *IEEE Trans. on Vehicular Technology*, vol. 59, no. 1, Jan. 2010.

[40] J. Cullum, R. A. Willoughby, "Computing eigenvalues of very large symmetric matrices – an implementation of a Lanczos algorithm with no reorthogonalization," *J. Comp. Phys.*, no. 44, 1981.

[41] M. Wax and T. Kailath, "Detection of Signals by Information Theoretic Criteria," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 33, no. 2, Apr. 1985.

[42] S. Kritchman, B. Nadler, "Non-Parametric Detections of the Number of Signals: Hypothesis Testing and Random Matrix Theory," *IEEE Trans. on Signal Processing*, vol. 57, no. 10, Oct. 2009.